Context-based coding of bilevel images enhanced by digital straight line analysis

Aghito, Shankar Manuel; Forchhammer, Søren

Published in:
I E E E Transactions on Image Processing

Link to article, DOI:
10.1109/TIP.2006.875168

Publication date:
2006

Document Version
Publisher's PDF, also known as Version of record

Context-Based Coding of Bilevel Images Enhanced by Digital Straight Line Analysis

Shankar Manuel Aghito, *Member, IEEE*, and Søren Forchhammer, *Member, IEEE*

***Abstract*—A new efficient compression scheme for bilevel images containing locally straight edges is presented. This paper is especially focused on lossless (intra) coding of binary shapes for image and video objects, but other images with similar characteristics such as line drawings, layers of digital maps, or segmentation maps are also encoded efficiently. The algorithm is not targeted at document images with text, which can be coded efficiently with dictionary-based techniques as in JBIG2. The scheme is based on a local analysis of the digital straightness of the causal part of the object boundary, which is used in the context definition for arithmetic encoding. Tested on individual images of standard TV resolution binary shapes and the binary layers of a digital map, the proposed algorithm outperforms PWC, JBIG, JBIG2, and MPEG-4 CAE. On the binary shapes, the code lengths are reduced by 21%, 27%, 28%, and 41%, respectively. On the map layers, the reductions are 31%, 34%, 32%, and 64%, respectively. The algorithm is also more efficient on the test material than the state-of-the-art generic bilevel image coder free tree.**

***Index Terms*—Digital straight line segments, map coding, MPEG-4, object-based video coding, shape coding.**

I. Introduction

THIS paper presents a novel bilevel image coder aimed at bilevel shape images and line drawings. Encoders for bilevel images as JBIG [11] and JBIG2 [10], [12] achieve highly efficient results for documents containing text and possibly halftone. Both standards offer template-based coding. JBIG2 also provides techniques designed for text and halftone regions. Vector-based coding has been explored, e.g., for binary video object shapes, but in the MPEG-4 [13] specification a template-based technique was chosen. Template-based coding may be improved using more dynamic modeling. High performance is achieved by the free tree coder [20], but at the cost of increased complexity, e.g., due to performing multiple passes on the image to be coded.

To address classes of bilevel image material for which there is no special mode in JBIG2, a new model is presented for efficiently coding parts of bilevel images which (locally) can be described as *digital straight line segments* (DSLS). The model is integrated in a conventional template-based coding scheme providing a robust encoder. Coding is performed in one single pass in raster scan order, which is possible because the local analysis of the source geometry is executed on the fly, without requiring a preprocessing step of vectorizing the source data.

Manuscript received January 25, 2005; revised August 9, 2005. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Amir Said.

The authors are with the Research Center COM, Technical University of Denmark, DK-2800 Lyngby, Denmark.

Digital Object Identifier 10.1109/TIP.2006.875168

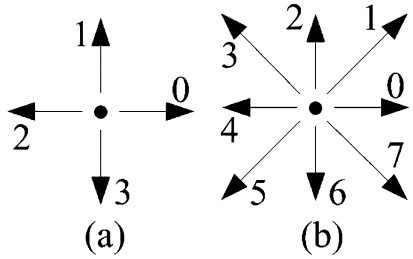

Fig. 1. Chain code elements; (a) 4-connected and (b) 8-connected sets.

The proposed model provides very efficient coding of binary shapes for image and video objects, binary layers of digital maps, line drawings, segmentation maps, and similar material. The work presented here is primarily optimized for coding binary shape images. The proposed encoder clearly outperforms the simple shape coding scheme (CAE) specified in MPEG-4 object-based video. A recent rate control analysis of MPEG-4 video object coding showed that the shape information has high priority in terms of operational rate-distortion [27]. The new H.264 Advanced Video Coder has greatly improved compression of texture information (luminance and chrominance) compared to previous video coding standards. Hence, more efficient shape coding is of interest for future object-based video encoders.

In Section II, digital straight line segments are presented and some of the known properties of ideal DSLS are reviewed. Based on these properties, the coding of ideal DSLS is treated and analyzed. These results are exploited in the development of a new coding scheme, described in detail in Section III in a version targeted at lossless intracoding of binary shapes for video objects. The complexity of the algorithm and the possible complexity reductions are considered in Section IV. Lossy coding based on preprocessing is discussed in Section V. Coding results for the suggested applications are presented in Section VI.

II. Digital Straight Line Segments (DSLS)

Object boundaries and lines contain most of the information in the types of binary image targeted by this paper. In this section, digital straight lines are introduced and the coding of perfect digital straight lines is considered.

A. Properties of DSLS

A boundary may be univocally described in terms of a 4-connected or an 8-connected set of edges (Fig. 1). Without loss of generality, we consider the 8-connected case in the first octant based on the digitization of the straight line

$$y(x) = \alpha x + e, \quad 0 \le \alpha < 1 \tag{1}$$

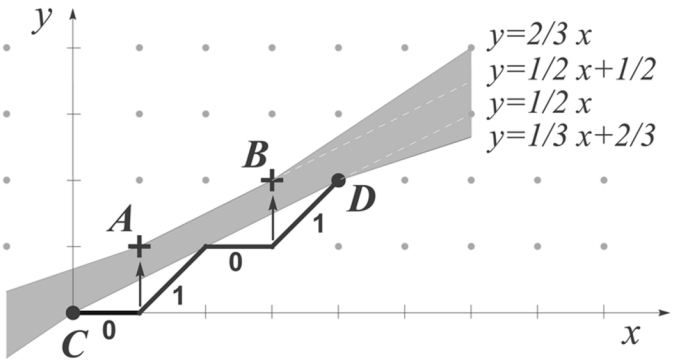

Fig. 2. DSLS with chain code 0101 and its (x, y) preimage (gray area).

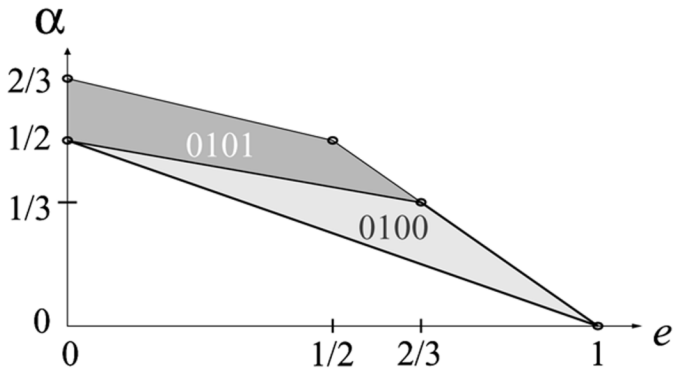

Fig. 3. (α, e) preimage of two DSLS with chain code 0101 and 0100. The union forms the preimage of the DSLS with chain code 010.

where α indicates the slope (angle of the line with respect to the x axis), and e indicates the vertical offset for $x = 0$. A DSLS with l elements, named d^l, is defined by the set of grid points (x_i, y_i) resulting from the digitization of (1) described by

$$y_i = \lfloor \alpha i + e \rfloor, \quad x_i = i, \quad \forall i \in \{0, 1, \ldots, l\}. \tag{2}$$

The $l + 1$ grid points (x_i, y_i) are 8-connected by l chain elements, $d_i, \forall i \in \{1, 2, \ldots, l\}$, each defined as 0 (if $y_i = y_{i-1}$) or 1 (if $y_i = y_{i-1} + 1$). Two important properties of such DSLS are derived in [7]. The first one is the mathematical definition of the preimage of the DSLS in the space defined by the parameters (α, e), which we describe in this section. Secondly, it is proven that every DSLS chain code can be represented by a set of four unique (nonnegative) integer parameters, none of which is greater than l.

In his classic work on chain codes, Freeman stated three properties of the chain code of digital straight lines (cited from [25]): "(F1) at most two types of elements can be present, and these can differ only by unity, modulo eight; (F2) one of the two element values always occurs singly; (F3) successive occurrences of the element occurring singly are as uniformly spaced as possible." For example, 012012 is not a DSLS since three distinct values are present (0,1,2) and likewise 0202 is not a DSLS because 0 and 2 differ by more than 1; the element 0 in the chain code 01101110 always occurs singly, while 011001110 cannot be a DSLS since both the values 0 and 1 occur in runs; the chain code 011011110 is not a DSLS because the runs of element 0 have length 2 and 4, which differ by more than 1. The last imprecise property has later been precisely formulated and many algorithms for easy recognition of a DSLS given its chain code are known. In this paper, the fast parsing algorithm proposed in [18] is used, as presented later in this section. A comprehensive overview of the properties of a DSLS is given in [25].

B. Preimages of DSLS

The preimage set of a given DSLS in the (x, y) plane is the set containing all the continuous straight lines which are digitized to this DSLS. There exist either three or four lines [7], called *bounding lines* (Fig. 2), which specify the limits of the preimage lines. Hereafter, we refer to the case with four bounding lines (Fig. 2), since the three-line case can be treated as a special case of the four-line case. The points (A, B, C and D) located at the intersections of the bounding lines are called *limiting points*. The slope and the offset with respect to the x axis of each bounding line may be calculated from these points. The slope is given by a rational number [7]. Let p and q denote relatively prime nonnegative integers not greater than l, with ratio p/q, equal to the slope of two parallel bounding lines defining the middle part of the preimage (given by the segments $\overline{AB}$ and $\overline{CD}$). Likewise, p^+, q^+, p^-, and q^- are found so that p^+/q^+ indicates the bounding line with the maximum slope (given by $\overline{CB}$), and p^-/q^- indicates the bounding line with the minimum slope (given by $\overline{AD}$). Each bounding line is bijectively mapped to a point in the (α, e) plane. The interior of the quadrangle or triangle with vertices in these points is the preimage of the DSLS in the (α, e) plane, as shown in Fig. 3.

C. Coding of DSLS

The DSLS properties may efficiently be used for coding. Before introducing the proposed scheme, other work on DSLS is shortly discussed. The order of the number of (8-connected) line segments of length l with fixed starting point was determined as l^3/π^2 [17], although no simple coding scheme was provided. Thus, in this case, the per symbol rate asymptotically approaches zero at a rate not greater than $(3 \log l)/l$. In another study [22], digitization of straight lines at all angles was used to analyze the rate for individual coding of chain elements, showing that the entropy due to the digitization is significant. Even with the application of a standard template of, say, 10–16 pixels [12], conventionally used in bilevel image coding, the entropy remains relatively high, as illustrated in Fig. 6. We are, therefore, interested in making use of the high-order structure of a DSLS in a feasible coding scheme.

1) Sequential Coding of A DSLS: Given a sequence of chain code elements d^L constituting a DSLS, we are interested in sequentially encoding the elements. The next element d_{l+1} shall be encoded conditional on the first l elements d^l for $0 \leq l < L$. Without loss of generality, it is assumed that $0 \leq \alpha < 1$. Let a and b denote the two possible values for each chain element. Given d^l, two situations are possible. A *unique extension* of the DSLS occurs if there is only one value for the next element d_{l+1} such that the extended chain code, $d^l d_{l+1} = d^{l+1}$, is also a DSLS. No information is required to encode these chain elements. A *nonunique extension* occurs when $d^l a$ and $d^l b$ are both DSLS. Nonunique extensions occur less frequently then unique

extensions, and their frequency decrease with increasing length l of the DSLS. For a nonunique extension, the (α, e) preimage of d^l is bisected in two regions corresponding to the two DSLS extensions $d^l a$ and $d^l b$ (Fig. 3). Given an integrable probability density function (pdf) on the (α, e) parameters, it is possible to determine the conditional probabilities.

Let $A(d^l a)$ and $A(d^l b)$ denote the areas of the two (α, e) preimages. Assuming a uniform pdf on (α, e) the conditional probabilities of the two chain values are given by the areas

$$P(a \,|\, d^l) = \frac{A(d^l a)}{A(d^l a) + A(d^l b)} \tag{3}$$

where $P(a \,|\, d^l) = 1 - P(b \,|\, d^l)$ denotes the probability of element a at position $l+1$ given d^l. Applying arithmetic coding to these conditional probabilities provides an optimal coding.

The (α, e) preimage is the union of two triangles having a base of width $1/q$, representing the maximum range of possible offsets e occurring for $\alpha = p/q$ [7]. Thus, the area is

$$A(d^l a) = \frac{1}{2q(a)} \left(\frac{p^+(a)}{q^+(a)} - \frac{p^-(a)}{q^-(a)} \right) \tag{4}$$

where the argument (a) refers to the DSLS preimage of $d^l a$. Inserting (4) into (3), for both $d_{l+1} = a$ and $d_{l+1} = b$, gives the conditional probability. A simplified approximation is given by just considering the ratio of the q values in (4)

$$\hat{P}(a \,|\, d^l) = \frac{q(b)}{q(a) + q(b)}. \tag{5}$$

Note that there is a one-to-one mapping between the limiting points of the parsing algorithm described in Section II-D, which is formulated for the 4-connected case, and the edges of the preimage domain. Therefore, it is possible to calculate the probabilities (3) also for the 4-connected case.

2) Application of LZ78 Coding to a DSLS: Even without assuming that the sequence of chain elements is a DSLS, the analysis below shows that efficient coding is possible. Consider a digital straight line specified by given slope α and offset e (in a given octant). For 8-connected elements, the number of different DSLS of length k is (at most) $k+1$ [7]. So, obviously, the rate should converge to zero asymptotically. A simple analysis is here carried out for the classical LZ78 universal coding scheme [29] applied to the binary sequence of chain elements. A scenario providing an upper bound on the per symbol code length is provided by assuming that for all k, all $k+1$ distinct DSLS of length k are entered into the dictionary before a string of length $k+1$ is entered. In this case the number of elements parsed when all the DSLS strings up to length k are entered in the dictionary is

$$l(k) = \sum_{i=2}^{k} i(i+1) = \frac{1}{3}k^3 + k^2 + \frac{2}{3}k - 2 > \frac{k^3}{3} \tag{6}$$

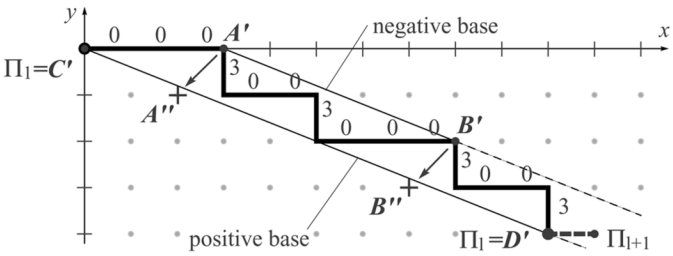

Fig. 4. Notation for the DSLS parsing algorithm [15].

where the inequality holds for any integer $k > 1$. The corresponding number of entries when all the DSLS strings up to length i are entered in the dictionary is

$$N(i) = \sum_{j=1}^{i} j + 1 = \frac{(i+2)(i+1) - 2}{2}. \tag{7}$$

Combining these equations, the code length for the entries is bounded by

$$\begin{aligned} \mathcal{L}(k) &\le \sum_{i=2}^{k} (i+1)(\log(N(i)) + 2) \\ &= 2\sum_{i=2}^{k} i\log(i) + o(i\log(i)). \end{aligned} \tag{8}$$

Evaluating the sums (7)–(8) by integrals, it is seen that the upper bound on the code length increases as $k^2 \log k$ for $k \to \infty$, and, therefore, the upper bound for the rate $\frac{\mathcal{L}(k)}{l(k)}$ converges to 0 for $l \to \infty$ as

$$\frac{k^2 \log k}{l} < \frac{\log l}{(3l)^{1/3}}. \tag{9}$$

This convergence (of the upper bound) is slower than what was obtained based on the number of DSLS lines, but the coding is sequential and without the use of knowledge of the DSLS properties in the coding process. Furthermore, it may be noted that a DSLS has infinite memory, whereas the universality of LZ78 was proven under the assumption of a finite order model.

D. Parsing a DSLS

A simple and fast algorithm for sequentially parsing or recognizing a DSLS from a sequence of chain elements shall be used. The algorithm was proposed in [18], but, in this paper, a notation similar to the one in [15] is used. The algorithm is formulated for the 4-connected edge description. It is based on the calculation of the *positive base* and the *negative base*, defined as the two parallel lines with the minimum distance, such that the sequence of chain elements is contained between the two lines. The two bases are determined by four points, A', B', C', and D'. An example is shown in Fig. 4 for the DSLS with chain code 00030030003003. A chain code containing only one distinct value is obviously a DSLS, so the recognition algorithm is initialized when an element with the second distinct value occurs.

Let t denote the index of this element, i.e., $t = 4$ in the example of Fig. 4. Without loss of generality, we can always replace the first element with a 0 and the second distinct element with a 3 (obviously a given chain code is a DSLS if and only if the chain code obtained with this substitution is a DSLS). The four points are initialized to $C' = (0,0)$, $A' = B' = (t-1,0)$, and $D' = (t-1,-1)$. At this point, it is still certain that the chain code is a DSLS. The recognition process for the following chain elements is then carried out recursively. Assuming that a sequence of $l+1$ (4-connected) grid points $\Pi_i, \forall i \in \{0,1,\ldots,l\}$ forms a DSLS of length l, we want to know if the chain obtained adding a new grid point $\Pi_{l+1} = (x_{l+1}, y_{l+1})$ is still a DSLS. The information about the given DSLS is, besides the length, entirely captured by the four points [7]. Let (u,v) be the vector parallel to the positive and negative bases, with relatively prime integer coordinates. Introduce $w = uy - vx$ for any point (x,y) on the negative base. In the example of Fig. 4, $(u,v) = (5,-2)$ and $w = 6$. For the new grid point, calculate $h(l+1) = vx_{l+1} - uy_{l+1} + w$. There are four possible cases, described below, where the subscript OLD indicates the parameters for the given DSLS (e.g., calculated in the previous step of the recursion).

1) $0 \le h(l+1) \le |u| + |v| - 1 : \Pi_{l+1}$ is on the given DSLS.
2) $h(l+1) = -1$: The $l+1$ vertices form a new DSLS delimited by $B' = \Pi_{l+1}, A' = A'_{\mathrm{OLD}}, C' = D' = D'_{\mathrm{OLD}}$.
3) $h(l+1) = |u| + |v|$: The $l+1$ vertices form a new DSLS delimited by $D' = \Pi_{l+1}, C' = C'_{\mathrm{OLD}}, A' = B' = B'_{\mathrm{OLD}}$.
4) Otherwise, the $l+1$ vertices do not form a DSLS.

Hence, in the first three cases, the final chain code is a DSLS, and a new grid point could be inserted, while in the last case the algorithm stops, concluding that only the first l points form a DSLS. In the example of Fig. 4, the new grid point $\Pi_{l+1} = (11,-4)$ gives $0 \le h(l+1) = 4 \le 5 + 2 - 1$. Hence, this situation falls into case 1) and the extended chain code 000300300030030 is a DSLS. This recognition algorithm is easy to implement and fast, requiring only integer arithmetic and on the average about ten operations per point. The recursive structure and the fact that all the information about the DSLS is captured in the four limiting points are two nice properties that can be used to simplify the implementation of the encoder proposed in the next section. Assuming $|v| < u$, the limiting points of the preimage given by A, B, C, and D are obtained by mirroring the corresponding points A', B'', C'', and D' with respect to the x axis (Fig. 4). Hence, $q = u, p^+/q^+$ and p^-/q^- are known, providing all the information required in (4) to calculate the area of the (α, e) preimage.

III. Lossless DSLS-Based Coding

This section presents the new algorithm, the DSLS-based coder (DSLSC), in a version targeted at binary shapes for image and video objects. Only the intracase is considered, i.e., temporal correlation between successive images is not exploited.

Binary shapes for object-based video mainly consist of large connected uniform regions. Small isolated groups of pixels or thin lines do not occur frequently. The information to be utilized for efficient coding lies along the boundary of the objects. The task of the encoder may be viewed as predicting whether pixels lie before or after the boundary, which is typically continuous and locally representable by a DSLS, as shown in Fig. 6.

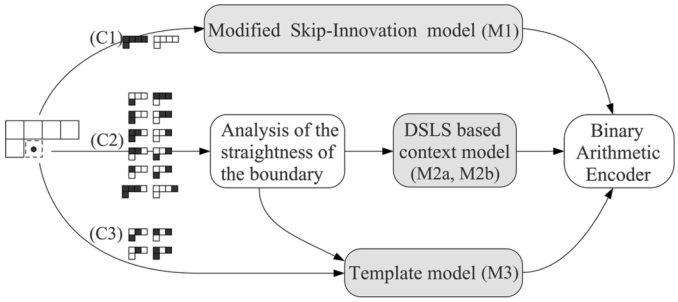

Fig. 5. Architecture of the proposed coding scheme. The models used for coding are highlighted.

The high-level architecture of the encoder is presented below, followed by a detailed description of each block. Extensions and modifications for improved performance for intercoding and other suggested applications are discussed in Section VI.

Conventional adaptive template coding is suboptimal because the context provided by the template does not eliminate uncertainty of pixels along the boundary, even when this is an ideal DSLS (Fig. 6). Using a larger template is not an efficient way to address this problem. Since the number of contexts grows exponentially with the size of the template, accurate estimation of the coding probabilities is only possible if the template is rather small, compared to the structures of the DSLS. This is especially the case if the orientation of the boundary is close to the orientation of the x or the y axis.

A. Encoder Architecture

The long-term dependencies in the DSLS can be exploited, reducing the uncertainty of boundary pixels, using the theoretical framework presented in Section II. For practical image coding, we cannot rely on having perfectly straight lines, but the theory provides a good context for coding the pixels in proximity of a boundary. The basic principle is that, if the causal part of a boundary is straight, then there is a high probability that locally the boundary will continue straight in the same direction.

The simple high-level architecture of the proposed coding scheme is shown in Fig. 5. The pixels are encoded sequentially in raster scan order. When a new pixel is to be encoded, three cases are distinguished, based on a five-pixel template: the *uniform region* (C1); the *object boundary* (C2); and the *irregular structure* (C3) (the configuration of the template pixels determines the case, as specified on Fig. 5). Specialized coding models are selected for each situation. The *modified skip-innovation model* $(\mathrm{MSI} = \mathrm{M1})$ is used to encode uniform regions efficiently. Object boundaries are processed by the analysis block. If the information provided by the boundary is reliable the *DSLS model* (M2) is utilized to predict the probabilities for the value of the current pixel. *Template coding* (M3) is used for the remaining pixels. The DSLS and the template models encode one single pixel at the time, while the MSI model encodes a run of contiguous pixels. The coded bitstream is generated in one single raster order pass, feeding the probabilities estimated by the selected model to one single binary arithmetic encoder, implemented as described in [23].

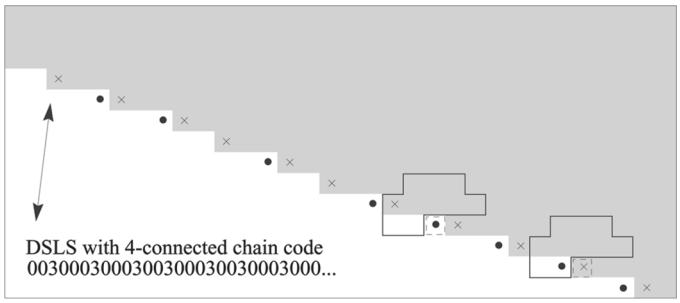

Fig. 6. Excerpt of the binary shape sequence *akiyo* (extracted from natural video). Though all of the marked pixels has the same causal template, pixels marked with (•) are within the object and pixels marked with (×) are in the background. Hence, the template cannot provide a good context for coding these pixels.

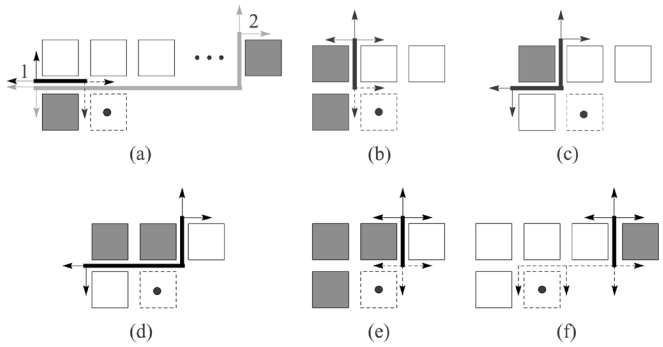

Fig. 7. Cases for DSLS analysis. The arrows indicate the directions for tracking the causal part of the edge and the allowed extensions around the current pixel •. For case (a), the direction 1 is tracked first, if there is not a DSLS then 2 is considered. Case (f) is not considered by default.

B. Analysis of the Straightness of the Boundary

If an object boundary is detected (C2), the analysis block is used to decide whether the local boundary can give reliable information about the current pixel, in which case the DSLS model (M2) is selected. The 4-connected boundary (Fig. 6) is analyzed as a 4-connected DSLS. The edges between black and white pixels constitute the chain elements. Let T^4 be the template with the four closest causal neighbors, ordered clockwise starting from the left of the coding pixel. The causal part of a boundary is tracked along the directions indicated in Fig. 7, using the fast algorithm described in Section II-D. The resulting DSLS is represented by the 4-connected chain code c^l, with elements $c_i, i \in \{1, 2, \ldots, l\}$, where l is the length of the DSLS, c_1 the most remote and c_l the most recent element. Let s be the value of the singular element as defined by (F2). Let n_s be the number of runs of nonsingular elements in c^l. If c_l is nonsingular, r_{cur} is the length of the current run and $r_{\min}$ is the length of the shortest among previous runs. The chain code c^l is defined to be *reliable* if and only if any of the statements (R1–R5) below is true (default setting). The additional statement (R6) may be included as an option, in which case the chain code is *reliable* if and only if any of (R1–R6) is true

$$
\begin{aligned}
&(\text{R1}) \quad n_s \geq n_{s,\min} \\
&(\text{R2}) \quad l \geq l_{\min} \\
&(\text{R3}) \quad (c_l = s) \bigwedge (l \geq l_{\min,s}) \\
&(\text{R4}) \quad (T^4 = z011 \bigvee z100) \bigwedge (2 \in c^l) \\
&\qquad \bigwedge \{(n_s > 1) \bigvee [(n_s > 0) \bigwedge (c_l = s)]\} \\
&(\text{R5}) \quad (T^4 = 0001 \bigvee 1110) \bigwedge (0 \in c^l) \\
&\qquad \bigwedge \{(n_s > 1) \bigvee [(n_s > 0) \bigwedge (c_l = s)]\} \\
&(\text{R6}) \quad (c_l \neq s) \bigwedge (r_{\text{cur}} < r_{\min}/2)
\end{aligned}
\tag{10}
$$

where z is either 0 or 1 and $l_{\min}, n_{s,\min}$, and $l_{\min,s}$ are parameters. Small values of these thresholds are optimal for shapes with perfectly straight boundaries, larger values are more suitable for noisy or irregular edges. The statements (R1) and (R2) are the most important and have a straightforward interpretation. Suppose, for example, $l_{\min} = 10$ and $n_{s,\min} = 3$. In this case, the DSLS with chain code 01010010 is reliable since $n_s = 4 \geq n_{s,\min}$. The chain code 0000010000 is reliable because $l = 10 \geq l_{\min}$. On the other hand, 0100 is not reliable, since $n_s = 2$ and $l = 4$ are smaller than the respective thresholds. It is intuitive that although this chain code is a DSLS, it is very short and could easily occur in irregular portions of the boundary, in which case the DSLS model would probably fail. Fairly good results were obtained in [3] using only (R2). The last four statements enable the use of the DSLS model in more specific situations, because in the target material a certain regularity of the boundary is still expected even when (R1) and (R2) fail. Suppose $l_{\min,s} = 3$. The chain code 0001 is reliable since $c_l = s$ and $l = 4 \geq l_{\min,s}$; hence, (R3) is true. Intuitively, if the boundary is regular, there is a higher probability of the next element being a 0, in which case the DSLS model gives the right prediction, since 00010 is a DSLS while 00011 is not. Suppose now $T^4 = 0011$ [as in Fig. 7(b)] and $c^l = 3233$; although none of (R1-R3) is true, c^l is reliable because (R4) is satisfied. In fact, unless the edge is strongly irregular, it is unlikely that the next element is 0. Accordingly, the DSLS model would assign a small probability to this event, since 32330 is not a DSLS. A similar example could also be made for (R5). Finally, for the DSLS with chain code 000010, $r_{\text{cur}} = 1$ and $r_{\min} = 4$, so (R6) is true. The DSLS model would predict the next element being a 0, which is reasonable for a regular boundary.

The DSLS is only tracked as long as $l \leq l_{\max}$ (default value 50) and $n_s \leq n_{s,\max}$ (default value 15).

The DSLS model (M2) distinguishes between deterministic (M2a) and nondeterministic (M2b) decisions.

C. Coding Deterministic Decisions With the DSLS Model

The DSLS parsing algorithm is used to determine the valid values of c_{l+1}, i.e., the values for which $c^l c_{l+1}$ is a DSLS. Assume that the actual boundary continues as a DSLS. If for any valid c_{l+1}, the current pixel is on the same side of the boundary, then the value of the current pixel is deterministic. This is, of course, the case for a unique extension, but it can also be true when two values of c_{l+1} are allowed if both cases lead to the same value for the current pixel. Suppose for example $T^4 = 0001$ [as in Fig. 7(e)] and $c_l = 0333033$. The next element c_{l+1} can be either 0 or 3. In both cases, the coding pixel lies to the left of the extended boundary c^{l+1}, so the coding pixel

is deterministically 0. For practical coding, this value provides a prediction of the current pixel. The information to be coded is whether this prediction is right or wrong (M2a). A *right deterministic decision* r or a *wrong deterministic decision* w is encoded with the adaptively estimated probabilities

$$P(r) = \frac{n_r + \delta_r}{n_r + n_w + \delta_r + \delta_w} \quad \text{and} \quad P(w) = 1 - P(r) \tag{11}$$

where n_r and n_w are the numbers of right and wrong decisions encoded so far, and the parameters (δ_r, δ_w) are set to $(9, 1)$, giving $P(r) = 0.9$ as the initial value.

The context for the statistics is determined by the value of n_s (quantized in $I_{\rm ns}$ logarithmically spaced intervals), the template T^4 (Fig. 7) where each case can either be separated or lumped together with its inverted configuration, the value of $r_{\rm cur}$ (quantized in $I_{\rm rc}$ intervals between 0 and the maximum length of the runs of nonsingular elements in c^l), and whether there is a unique extension or not.

D. Coding Nondeterministic Decisions With the DSLS Model

For nonunique extensions, a deterministic decision cannot be made if both values of c_{l+1} correspond to a distinct value for the coding pixel. A probability may be assigned to each case based on preimages in the (α, e) plane, as described in Section II-C. If (R1) is true, i.e., $n_s \geq n_{s,\min}$, the pixel value is encoded (M2b) with probabilities proportional to the area of the preimage of the corresponding DSLS $c^l c_{l+1}$, using (3)–(4) (default choice) or (5). If (R1) is not true, the template coding mode (M3) is applied.

E. Modified Skip-Innovation Model

If a uniform region is detected, a modified skip-innovation model (M1) is applied. Skip-innovation (SI) codes were introduced in PWC [5] as a fast and efficient way for coding large uniform regions. When a uniform template is found, there is a high probability that a run of pixels with the same value will occur. We apply a five-pixel template, whereas in PWC, four pixels are used. The corresponding run length S in the previous row gives an estimate of the actual run length I. The SI code for I is either a single 1 indicating a full skip ($I \geq S$), or a 0 indicating $I < S$ followed by a binary representation of I. SI code words are then entropy coded using context-based arithmetic encoding.

We propose a modification based on modeling the situations called pseudoinnovations in [5] (Fig. 8). The goal is to increase the probability of full skips by assuming that the edge of the structure occurring at the end of the run on the line above is a DSLS. As illustrated in the figure, analyzing the DSLS a run length shorter than S is highly probable. Let S'_D be the longest run allowed by all the possible straight extensions of the DSLS to the current row. If the causal part of the DSLS is not shorter than the specified value of $l_{\min,SI}$ (default value 3), then S is replaced by $S_D = \max\{S'_D, S - r_{\max}\}$ (by default $r_{\max} = 4$). The lower bound $S - r_{\max}$ prevents a too large reduction of S, which might not be desired if the edge is actually not straight.

The binary representation of I assigns the shortest codes to the longest runs [5], recognizing that pseudoinnovations are more likely to happen in the proximity of a full skip. Small values of I are also common, as pseudoinnovations occur frequently also in the proximity of the beginning of the run. Thus, we propose an alternative representation of I which uses $N_{\rm bs}$ bits (default value 5) for the smallest and the largest values, and longer variable length codes for the intermediate values. This representation is only used if $S_D > 2^{N_{\rm bs}}$. The first bit indicates if the value of I is one of the $2^{N_{\rm bs}-2}$ smallest or the $2^{N_{\rm bs}-2}$ largest allowed values. If this is the case, then the remaining $N_{\rm bs} - 1$ bits specify the value of I, otherwise the binary representation for I adopted in PWC is used (properly shifting the range, because some of the values are already treated in the previous case).

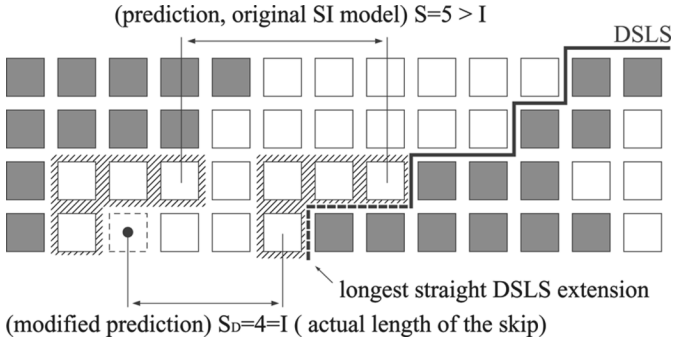

Fig. 8. Using the basis SI model ($I < S$) a failed full skip followed by a code for I is required. Analyzing the boundary we can reduce the predicted run length to $S_D = I$ efficiently coded as a single full skip.

F. Template Coding

Template-based coding (M3) is utilized as a last resource, to encode the remaining pixels that were not coded by one of the algorithms described above. The template is formed with $|T|$ causal neighbors of the coding pixel, ordered with increasing Euclidean distance. The coding context is augmented keeping separate statistics for the following three cases considered in order, i.e., each case is only considered if none of the previous cases occur.

1) The template T^4 indicates an irregular structure (C3).
2) The detected object boundary (C2) is not reliable.
3) A nonunique extension and $n_s < n_{s,\min}$, or $c^l c_{l+1}$ is a DSLS for three distinct values of c_{l+1}.

The last case 3) is a special case for nonunique extensions. Conventional techniques are used for adaptivity. The counts are updated after each coded pixel and scaled periodically. Generally, the number of pixels encoded with this model is small (Fig. 9) and their uncertainty is high, and it is not useful to use a large template for encoding. This indicates that most of the useful information is captured by the DSLS and the MSI models, and the remaining pixels are fairly random.

IV. Fast Implementation and Fuzzy Versions

The work of this paper has focused on high compression performance and the DSLS theory, but fast implementation should be achievable. Using MSI provides fast coding for most of the pixels. For the targeted material, the majority of the pixels are coded as full skips or long innovations. The DSLSC can be implemented as a single pass raster order process. The DSLS

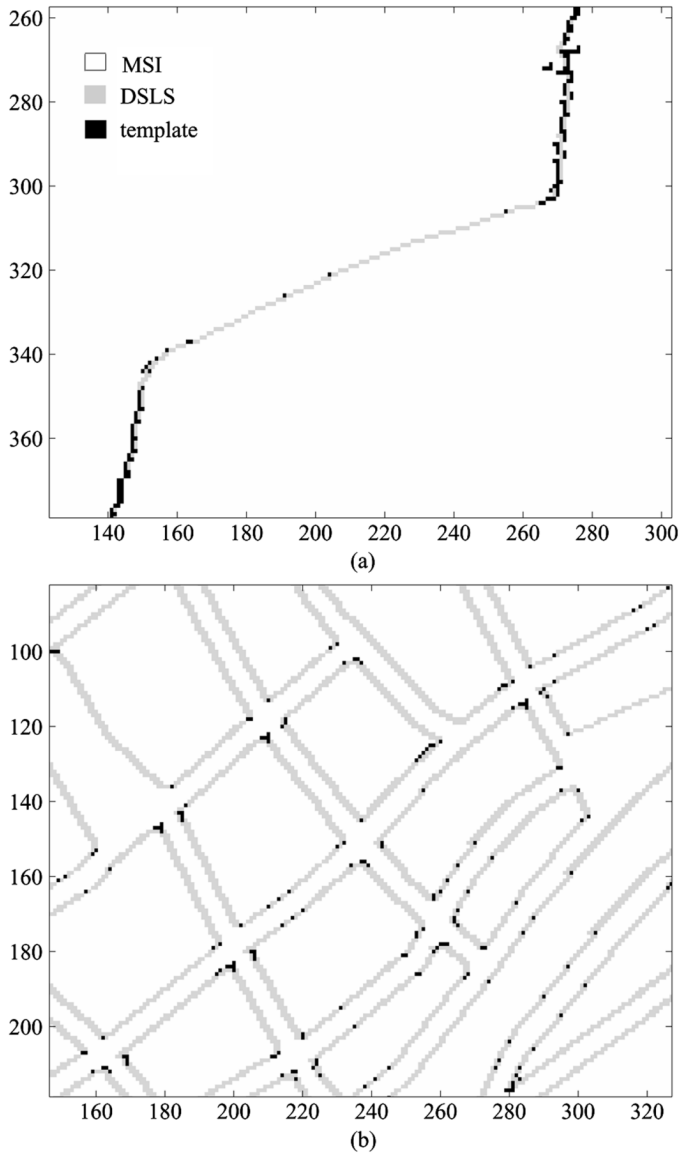

Fig. 9. Use of the different coding modes for (a) the first frame of the sequence *akiyo* and for (b) the layer *road white* of the map of Copenhagen.

parsing algorithm utilized is sequential, meaning that each time a section of a boundary is tracked and modeled as a DSLS, the few parameters defining the DSLS (Section II-D) can be stored and passed on to the following row, making the process quite fast. Internally in the encoder and decoder, the edges could be stored using run lengths to the next edge and the parameters of the edge. Storing these parameters is not a problem, because the typical binary target images do not contain a large number of boundaries. For bilevel images with many boundaries, e.g., text documents, the DSLS coding mode is generally not competitive and should be disabled.

Many variations of DSLSC may be realized, the main principle being to track the causal boundaries for prediction. Fuzzy versions could be implemented relaxing the necessary conditions for a DSLS, e.g., only using Freeman's first two conditions $(\mathrm{F1}+\mathrm{F2})$, which would make the implementation simpler, and eventually more suitable for a larger class of images with a lower proportion of perfect digital straightness.

V. Lossy Coding Based on Preprocessing

If loss of information is permitted, significant reductions in code length may be achieved with DSLSC. In this paper, we do not present one specific algorithm for preprocessing of data (hereafter named filtering) targeted at DSLSC, but we demonstrate that the proposed encoder can take full advantage of using different types of filters, although these were designed for other image compression schemes. The reason for this is that these filters generally reduce the amount of noise and irregularities in the image, which directly or indirectly leads to more and longer straight edges. Different filters are described here. The rate distortion performance obtained combining the described filters with JBIG, JBIG2, and DSLSC is presented in Section VI.

Simple filters, such as morphological filters and median filters, are often used for noise reduction, but if the data is not noisy, this type of filtering should generally be avoided.

A more intelligent solution is the *feature-based filter* proposed in [8] for line-drawing images. The visually lossless filtered image is obtained by flipping isolated groups of mismatch pixels between the original image and a vectorized and simplified version of the original image.

Another possibility is the *rate distortion flipping* proposed in [21]. For a given adaptive template-based encoder, e.g., JBIG, the effect on the overall code length due to changing the value of a single pixel can be calculated from the statistics of the given image. Thus, a rate distortion tradeoff can be realized by greedily flipping the pixels which produce the largest rate reduction.

A. Polygonal Approximation for Shape Coding

Context-based arithmetic encoding (CAE) similar to JBIG was adopted in MPEG-4 for coding binary shapes. A vertex-based encoder was also investigated, based on differential encoding of the vertices forming the polygonal approximation of the shape contour. This representation can be very efficient, especially at high distortions. CAE is more robust, simpler to implement, and more efficient in intermode. The performance of the two algorithms was different in different situations, though.

DSLSC combined with polygonal approximation can close the gap maintaining the good properties of both methods. Straight boundaries produced by the polygonal approximation are encoded very efficiently, and template coding is used for irregular edges. The raster order single scan structure of CAE is maintained, which simplifies implementation. The template coding mode can be extended to the intercase in the same manner as CAE.

We evaluated DSLSC with a simple polygonal approximation scheme. The contour is first losslessly represented as a polygon with vertices $V_i, \forall i \in \{1, 2, \ldots, N_{\mathrm{V}}\}$. Vertices are ordered counterclockwise starting from V_1, found as the first pixel within the shape encountered scanning the image columnwise. The contour is closed, so $V_1 = V_{N_V}$. The approximation is obtained scanning vertices in the given order, replacing two successive segments $\overline{V_iV_{i+1}}$ and $\overline{V_{i+1}V_{i+2}}$ with $\overline{V_iV_{i+2}}$ if the mean distance from this segment to the corresponding portion of the original boundary is within a certain value d_{M}. The algorithm is iterated for increasing integers j until $d_{\mathrm{M}} = j/10$ reaches the selected distortion d_{MAX}.

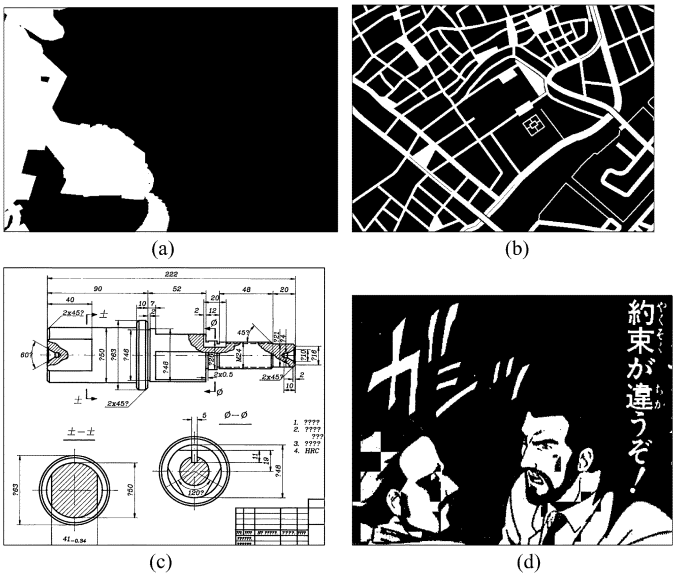

Fig. 10. (a) Target material for the proposed encoder. The first frame of the sequence *robot*. (b) The *road white* layer of the map of Copenhagen. (c) The image *demo2* from the set of engineering drawings. (d) The foreground/background separation map generated by SLIm [26] on a comic book image.

Our purpose here is not to achieve the best rate distortion performance, but rather to demonstrate how polygonization provides an efficient preprocessing step for our coding scheme. This is clearly shown in Fig. 11. Performance could be improved since this polygonization is not optimal. The vertices of the approximation are forced to belong to the original boundary, while it is known that rate distortion performance is greatly improved if these vertices can be placed in proximity of the original boundary [14] [19]. A better choice of the initial point V_1 may also improve the polygonization.

VI. Applications and Results

In this section, the efficiency of DSLSC is evaluated presenting results for binary shape images of video object sequences as well as binary layers of digital maps, line drawings, and segmentation maps for region-based image coding. The encoding parameters are optimized for each application. A simpler version of DSLSC was presented in [3], where fewer parameters were utilized achieving almost the same performances reported in this paper. Examples of target images are shown in Fig. 10. The following encoders are tested for comparison. Markus Kuhn's implementation of JBIG [11] (available at http://www.cl.cam.ac.uk/~mgk25/jbigkit/), the generic part of JBIG2 (using the MQ coder with 10–16 pixels standard templates), PWC [5] (available at http://www.caravian.com/), CAE (MoMuSys MPEG-4 reference software), the bilevel encoder included in the open source implementation of DjVu (http://djvulibre.djvuzone.org), and, finally, the free tree coder [20], which is a multipass algorithm providing to our knowledge the best published coding results on general purpose bilevel images. CAE is tested both in intra- and intermode, while all the other algorithms do not make any use of temporal correlation between successive frames in the sequence. A small contribution for header information (image size and coding parameters) should be added for DSLSC, CAE, and the free tree coder (the cost for coding the tree is already included). Only the binary case is treated in this paper, but a multilevel version of DSLSC could easily be realized.

TABLE I
Lossless Code Lengths (Bytes/Frame) for Standard TV Resolution Binary Shape Sequences

seq., layer, frames	DSLSC_1	Free Tree	PWC	JBIG	JBIG2 (16)	CAE intra	CAE inter
akiyo, 1, 1-300	**211**	229	290	354	331	385	258
weather, 1, 1-300	**261**	273	356	412	403	425	260
bream, 1, 1-300	**381**	397	556	571	543	964	538
sean, 1, 2-301	**361**	366	436	494	493	569	476
news, 3, 1-300	**526**	541	639	683	685	752	228
coastg., 1, 1-300	**282**	287	335	346	364	348	94
stefan, 1, 1-300	144	**129**	173	182	189	226	210
child., 1-kids, 1-300	674	**660**	753	829	859	906	731
td, 1, 11-124	**207**	218	262	286	295	237	228
td, 2, 1-78	**38**	52	82	86	84	41	4
td, 3, 8-131	565	**507**	781	810	844	1213	1157
td, 4, 1-78	**110**	118	143	160	161	121	122
td, 5-robot, 1-131	**390**	435	505	530	534	563	468
td, 6-explos., 1-29	**79**	93	134	136	125	310	300
td, 7, 1-105	**233**	247	295	324	338	262	170
td, 8-rain, 1-169	**1643**	1695	2227	2254	2420	3380	3167
td, 9, 1-15	677	**634**	715	791	831	871	876
td, 10, 1-64	736	**685**	776	848	887	949	934
average	**414**	421	527	565	576	701	518

A. Object-Based Video Coding

DSLSC as presented in this paper is very efficient for independent encoding of single frames of binary shape sequences.

Coding results are presented in Table I for standard TV resolution sequences (720×486 pixels, 30 Hz). The first frame of the sequences *akiyo*, *kids*, *robot*, and *rain* were used as the training set to optimize the parameters, giving the default settings described in Section III and $n_{s,\min} = 3$, $l_{\min} = 25$, $l_{\min,s} = 7$, $I_{\mathrm{ns}} = 2$, $I_{\mathrm{rc}} = 6$, and $|T| = 6$. The encoder with this configuration is referred to as DSLSC_1. The interlaced sequences with fast motion *kids*, *weather*, *sean*, *bream*, and layers nine and ten of the sequence *total destruction* (*td*) were all deinterlaced prior to coding, simply by reading first the odd and then the even lines. Clearly, DSLSC_1 outperforms all the other algorithms on this test set of TV resolution sequences. The average coding gain is 21%, 27%, and 28% over PWC, JBIG, and JBIG2, respectively. DSLSC_1 achieves a 41% smaller code length than that obtained with CAE in intramode. Although DSLSC_1 does not exploit the temporal redundancy in the sequences, it produces 20% smaller code length than CAE in intermode. DSLSC_1 is also more efficient than the free tree.

The code lengths obtained by DSLSC_1 on the first (non empty) frame of each sequence in Table I was analyzed. About 98.5% of the pixels are quickly and efficiently encoded using the MSI model, 0.74 with the conventional template model, and 0.71% with the DSLS model. The different models contribute 35%, 43%, and 22% of the total rate, respectively. In particular,

TABLE II
LOSSLESS CODE LENGTHS (BYTES/FRAME) FOR LOW-RESOLUTION BINARY SHAPE SEQUENCES. THE FIRST EIGHT SEQUENCES ARE AVAILABLE AT FTP://FTP.TNT.UNI-HANNOVER.DE/PUB/MPEG/MPEG4_MASKS/. THE LAST TWO ARE USED IN [28]

seq., format, frames	DSLSC_2	Free Tree	PWC	JBIG	JBIG2 (10)	JBIG2 (16)	CAE intra	CAE inter
kids, SIF, 1-100	**224**	234	280	304	282	317	255	*217*
weather, QCIF, 1-100	**60**	70	104	108	92	105	65	*40*
robot, SIF, 1-44	**295**	315	358	370	345	393	350	*286*
logo, SIF, 1-100	**334**	337	403	411	395	457	382	*115*
foreman, CIF, 1-207	**113**	131	178	199	173	187	184	168
stefan, SIF, 1-100	**84**	94	129	142	123	139	93	96
news, CIF, 1-300	**132**	140	194	209	191	200	196	*74*
cyclamen, SIF, 1-100	**381**	411	477	500	477	522	498	*345*
weather, QCIF, 1-100	**53**	63	97	100	85	96	59	*21*
cyclamen, CIF, 1-100	**434**	477	540	567	539	594	604	*314*
average	**186**	201	251	266	246	271	246	*147*

about 20 right deterministic decisions occur for every wrong deterministic decision. Nondeterministic decisions are used only for 0.03% of the pixels, contributing 3% of the total rate. In [3], the DSLSC was implemented in a simplified version, and tested on a subset of the standard TV resolution sequences employed in this paper. The results were comparable to those presented here. It was observed that if the DSLS model was disabled, the average code length increased by 15%; if both the DSLS and the MSI models were disabled, i.e., only the template coding mode was used, the average code length increased by 23%.

Results for lower resolution sequences including QCIF (176×144), SIF (352×240), and CIF (352×288) are presented in Table II. The first frame of *foreman CIF*, *kids SIF*, and *cyclamen SIF* were used as training set. Optimizing the parameters gave $n_{s,\min} = 4$, $l_{\min} = 12$, $l_{\min,s} = 5$, $r_{\max} = 5$, $I_{\mathrm{ns}} = 1$, $I_{\mathrm{rc}} = 6$, $l_{\max} = 16$, $n_{s,\max} = 7$, and $|T| = 4$. Further, the statement (R6) in (Section III-B) is also utilized, and the statistics for deterministic decisions are kept separate for the inverted cases of T^4. The other parameters are set to default values. This configuration of the encoder is referred to as DSLSC_2. Again, DSLSC coding achieves the best (intra) results, giving 24% smaller average code length compared with CAE intra, which in this case performs comparably to the adaptive template-based encoders. At low resolutions, CAE inter is, indeed, 21% better than (intra) DSLSC_2, because the motion vectors are small and easy to code.

It is possible to efficiently exploit temporal correlation in DSLSC [4]. The template coding mode of DSLSC can be modified in the same manner as CAE inter. The statistics gathered in the reference frame can be used as initialization when coding the current frame, as shown in [2]. The position of wrong deterministic decisions and pseudoinnovations in the reference frame can be used to improve the DSLS and the MSI models.

DSLSC_2 gives good results also when compared to other encoders specifically designed for shape coding. The vertex-based shape coder evaluated in MPEG-4 gives a 7.8% lower average rate than CAE as reported in [14]. DSLSC_2 is 14% more efficient for *robot SIF* and similar results are obtained for *weather QCIF* and *kids SIF*. The skeleton-based method proposed in [28] gives the same results as DSLSC_2 for *weather QCIF*, but for *cyclamen CIF* the code length obtained with DSLSC_2 is 25% smaller. For lossy coding, the rate distortion flipping and the polygonization algorithms described in Section V are used to preprocess the data. Filtered versions of the first frame of the sequence *akiyo* are encoded with JBIG, JBIG2, DSLSC_1, and DSLSC_3. The resulting rate distortion curves are shown in Fig. 11, where the distortion measure is the one adopted by MPEG-4

$$D_{\mathrm{MPEG-4}} = \frac{\text{number of error pixels}}{\text{number of pixels in the original shape}}. \tag{12}$$

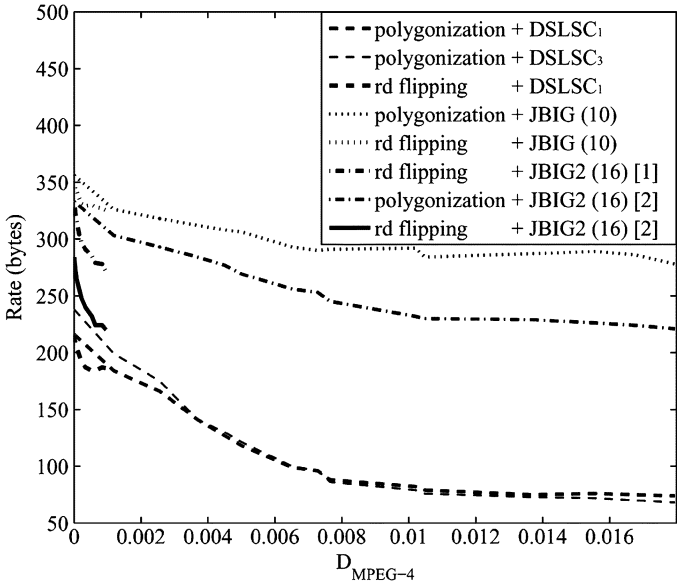

Fig. 11. Lossy coding of the binary shape of the first frame of *akiyo* by polygonization and greedy flipping of expensive pixels.

As expected DSLSC fully exploits the polygonal approximation. For DSLSC_1 the rate is reduced by up to 65% compared to the lossless case, while JBIG and JBIG2 achieve only 22% and 34% rate reductions, respectively. For the high distortion values DSLSC_1 only needs one third of the bits used by JBIG2. It is interesting to note that using DSLSC_3, optimized for the map of Copenhagen, the code length is 10% higher than DSLSC_1 in the lossless case but 8% lower for high distortions. This difference is small compared to the gain with respect to the other encoders, showing that the selection of the parameters is not so critical. Most of the rate reduction is already achieved at distortion values around 0.01.

The rate-distortion flipping [21] only allows limited changes of the image as shown by the short curve segments. The two JBIG2 implementations which use the standard 16-pixel template, for which the filtering is optimized, achieve about 60 bytes rate reduction, while for JBIG and DSLSC which both use a smaller template the reduction is about 30 bytes.

TABLE III
LOSSLESS CODING (BYTES) FOR THE LAYERS OF COPENHAGEN MAP

	$DSLSC_3$	Free Tree	PWC	JBIG	JBIG2 (10)	JBIG2 (16)	CAE
buildings	**604**	776	904	958	930	946	1310
land	**232**	320	416	445	416	444	763
water	**273**	360	471	495	466	492	882
fields	**74**	124	171	175	156	180	256
road yellow	**362**	436	624	658	630	631	1274
road white	2174	**2164**	3409	3570	3566	3281	5185
road pink	**212**	244	335	372	360	358	442
land cont.	405	**392**	672	725	704	537	1338
water cont.	468	**416**	503	511	498	508	1464
fields cont.	155	**148**	176	193	165	180	507
road cont.	3751	**3520**	4964	5098	5072	4621	10506
average	**792**	809	1150	1200	1178	1107	2175

B. Compression of Digital Maps and Line-Drawings

Digital street and topographic maps and line-drawings in general are likely to contain a considerable number of straight segments. Hence, DSLSC should also be very efficient on this type of image. The DSLSC was tested on 11 binary layers of a street map of Copenhagen [9] (723×546 pixels). Coding parameters are optimized for this set of layers, giving $|T| = 10$, $l_{\min} = 3$, $l_{\min,s} = 3$, $l_{\max} = 75$, $n_{s,\min} = 1$, $n_{s,\max} = 35$, $I_{\text{ns}} = 4$, $I_{\text{rc}} = 7$, the additional configuration in Fig. 7(f) is utilized, while the statement (R2) in (Section III-B) is not evaluated. The encoder with this configuration is referred to as DSLSC_3. Results are presented in Table III. DSLSC_3 is the best encoder for this map, 2% more efficient than the free tree, about 30% more efficient than JBIG, JBIG2, and PWC, and even 64% more efficient than CAE intra. CAE is of course not targeted at maps, but this demonstrates that the use of fixed probabilities is not a robust solution.

MSI is used for 96% of the pixels and is responsible for the 28% of the total rate. The DSLS model is used for 3.4% of the pixels, contributing 55% of the total rate. Template coding is used only for the remaining 0.6% of the pixels, contributing 17% of the total rate. More than 27% right deterministic decisions occur for every wrong deterministic decision. Nondeterministic decisions are used for 0.4% of the pixels, and contribute 21% of the total rate. For the simplified version of the DSLSC presented in [3], it was reported that by disabling the DSLS model, the average code length increased by 42%; if both the DSLS and the MSI models were disabled, the average code length increased by 56%.

Whereas DSLSC_3 is better than free tree coding on average, the latter is better than DSLSC_3 on the layers representing contours. The reason is that DSLSC as presented in this paper is mainly designed for binary object shapes, exploiting the properties described in Section III. The contour layers contain several thin lines which are not properly modeled by the current version of DSLSC. Preliminary experiments show that DSLSC can be extended in order to exploit also these structures, e.g., the code length for the *fields contour* layer has been reduced to 112 bytes.

TABLE IV
CODE LENGTHS (BYTES) FOR DIFFERENT TEST IMAGES. RESULTS FOR JBIG ON THE ENGINEERING DRAWINGS ARE FROM [8]. RESULTS FOR JBIG2 ON THE FINNISH MAPS ARE FROM [1]. K INDICATES 1024

	DSLSC	PWC	JBIG	JBIG2 (10)	JBIG2 (16)	DjVu
eng. draw., orig.	**26.9K**	30.1K	31.1K	30.4K	28.5K	28.7K
eng. draw., filt.	**22.9K**	27.3K	28.3K	28.3K	26.5K	26.2K
finn. maps, basic	1253K	1116K	1170K	1150K	**950K**	1159K
finn. maps, cont.	612K	616K	629K	630K	**550K**	639K
finn. maps, fields	**19.5K**	27.5K	28.4K	28K	26K	28.2K
finn. maps, water	**178K**	252K	264K	270K	220K	246K
for./back. (DjVu)	**748**	886	886	862	936	949
for./back. (SLIm)	**1996**	2098	2161	2159	2328	2253
CCITT	190K	183K	185K	184K	178K	**141K**

Using the same parameters optimized for the map of Copenhagen, DSLSC was tested on a set of 8 scanned engineering drawings [8] (up to 7296×4903 pixels) and on 16 binary layers of four Finnish topographical maps [1], [16] (5000×5000 pixels). Results are shown in Table IV. DSLSC_3 is the best encoder for most of the images. For the engineering drawings, DSLSC_3 is 11%–15% more efficient than the other encoders with the same template size. It is also 6% more efficient than JBIG2 with the standard 16-pixel template. Note that these images are scanned and the noise along the edges of the drawings distorts many of the straight lines.

Processing the line drawings with the perceptually lossless feature-based filter in Section V, DSLSC_3 achieves a 15% bit rate reduction, while the reduction for the other encoders is between 7%–10%. DSLSC_3 ($|T| = 10$) encodes the original drawings using 6% fewer bits than what JBIG2 (16-pixel template) requires to encode the filtered image.

On the *fields* layer of the Finnish topographical maps, DSLSC_3 gives 25%–31% smaller code lengths than the other encoders, and on the *water* layer the reduction is 20%–34%. On the layer *contours* DSLSC_3 is slightly more efficient than PWC, JBIG and JBIG2 with the ten-pixel template while it uses 11% more bits than JBIG2 with the 16-pixel template. On the *basic* layer, DSLSC_3 is less efficient than the other encoders, yielding 7%–32% larger code lengths. The relatively worse performance in the last two cases has the same explanation as for the contour layers of the Copenhagen map; hence, efficient coding should be achievable with a modified version of DSLSC capable of tracking thin lines. These images are also much larger than those in the training set, so increasing the number of coding contexts should give further improvements.

C. Segmentation Maps for Region-Based Coding

The concept of second generation image compression relies on a segmentation of the image in different connected regions that have different statistical properties. Different coding techniques or parameters can be used for each region. An efficient way to code the segmentation map is required. A related binary example is found in algorithms for coding mixed raster content images, utilizing a separation of (low frequency) background from a foreground with sharp edges [6]. Results for DSLSC_2 on

a segmentation produced by SLIm [26] and DjVu [6] are given in Table IV, illustrating the high efficiency of DSLSC also in this case.

The DSLSC_2 was finally tested on a set of eight CCITT images [20] (200 dpi, 1728×2339 pixels). The performance is not competitive compared to the other template-based coders, but fairly robust. The dictionary-based approach in DjVu outperforms all the template-based solutions. The DSLSC is not targeted at document images with text. The basic principle utilized in DSLSC could be generalized to develop new algorithms, more appropriate to this type of image. Besides digital straightness, the local curvature and average slope of the boundary could be used to improve the prediction of the current pixel.

VII. Conclusion

A new algorithm for coding bilevel images was presented. The algorithm is very efficient for coding (locally) digital straight edges in binary images. The high compression ratio is achieved by a new modeling strategy, based on the analysis of the causal boundaries of the object. When these can be modeled as digital straight line segments, an improved prediction of the current pixel is possible. The new model also includes a modified version of the skip-innovation coding of PWC. When the modeling is not deemed reliable, template-based coding is applied. The algorithm as presented in this paper is mainly targeted at binary shapes, although very efficient compression is achieved also for other classes of binary images such as layers of maps, engineering drawings or segmentation maps. Code lengths for test images within these image classes demonstrate the superior efficiency of the proposed encoder for most of the images. The algorithm may be optimized for specific classes of material and applications.

References

[1] E. I. Ageenko, P. Kopylov, and P. Fränti, "On the size and shape of multilevel context templates for compression of map images," in *Proc. IEEE Int. Conf. Image Processing*, Oct. 2001, pp. 458–461.

[2] S. M. Aghito and S. Forchhammer, "Context based coding of quantized alpha planes for video object," in *Proc. IEEE MMSP*, Dec. 2002, pp. 101–104.

[3] S. M. Aghito and S. Forchhammer, "Context based coding of binary shapes by object boundary straightness analysis," in *Proc. Data Comp. Conf.*, Mar. 2004, pp. 399–408.

[4] S. M. Aghito and S. Forchhammer, "Efficient Coding of Binary Shape Sequences for Object Based Video," in *Proc. Int. Workshop VLBV*, Sep. 2005, p. 4.

[5] P. J. Ausbeck Jr., "The piecewice-constant image model," *Proc. IEEE*, vol. 88, no. 11, pp. 1779–1789, Nov. 2000.

[6] L. Bottou, P. Haffner, P. G. Howard, P. Simard, Y. Bengio, and Y. LeCun, "High quality document image compression with DjVu," *J. Electron. Imag.*, vol. 7, no. 3, pp. 410–425, 1998.

[7] L. Dorst and W. M. Smeulders, "Discrete representation of straight lines," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-6, no. 4, pp. 450–463, Jul. 1984.

[8] P. Fränti, E. I. Ageenko, and A. Kolesnikov, "Vectorizing and feature-based filtering for line-drawing image compression," *Pattern Anal. Appl.*, vol. 2, pp. 285–291, 1991.

[9] S. Forchhammer and O. R. Jensen, "Content layer progressive coding of digital maps," *IEEE Trans. Image Process.*, vol. 11, no. 12, pp. 1349–1356, Dec. 2002.

[10] P. G. Howard, F. Kossentini, B. Martins, S. Forchhammer, and W. J. Rucklidge, "The emerging JBIG2 standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, no. 7, pp. 838–848, Nov. 1998.

[11] *Coded Representation of Picture and Audio Information–Progressive bi-level image compression*, ISO/IEC Int. Std. 11544, 1993.

[12] *Coded Representation of Picture and Audio Information–Lossy/Lossless coding of Bi-Level Images (JBIG2)*, ISO/IEC Int. Std. 14492, 2000.

[13] *Information Technology–Coding of Audio-Visual Objects–Part 2: Visual*, ISO/IEC Int. Std. 14496-2, 1999.

[14] A. K. Katsaggelos, L. P. Kondi, F. W. Meier, J. Ostermann, and G. M. Shuster, "MPEG-4 and rate-distortion-based shape-coding techniques," *Proc. IEEE*, vol. 86, no. 6, pp. 1126–1154, Jun. 1998.

[15] R. Klette and B. Yip, "The length of digital curves," *Mach. Graph. Vis.*, vol. 9, pp. 673–703, 2000.

[16] P. Kopylov and P. Fränti, "Compression of map images by multi-layer context tree modeling," *IEEE Trans. Image Process.*, vol. 14, no. 1, pp. 1–11, Jan. 2005.

[17] J. Koplowitz, M. Lindenbaum, and A. Bruckstein, "The number of digital straight lines on an NxN grid," *IEEE Trans. Inf. Theory*, vol. 36, no. 1, pp. 192–197, Jan. 1990.

[18] V. A. Kovalevsky, "New definition and fast recognition of digital straight segments," in *Proc. 10th Int. Conf. Pattern Recognition*, 1990, pp. 31–34.

[19] J. H. Lee, J. W. Chung, and J. K. Kim, "Optimal vertex adjustment method using a Viterbi algorithm for vertex-based shape coding," *Electron. Lett.*, vol. 35, no. 9, pp. 706–707, 1999.

[20] B. Martins and S. Forchhammer, "Tree coding of bilevel images," *IEEE Trans. Image Processing*, vol. 7, no. 4, pp. 517–528, Apr. 1998.

[21] B. Martins and S. Forchhammer, "Lossless, near-lossless, and refinement coding of bilevel images," *IEEE Trans. Image Process.*, vol. 8, no. 5, pp. 601–613, May 1999.

[22] D. L. Neuhoff and K. G. Castor, "A rate distortion analysis of chain codes for line drawings," *IEEE Trans. Inf. Theory*, vol. 31, no. 1, pp. 53–68, Jan. 1985.

[23] M. Nelson, *The Data Compression Book*. San Mateo, CA: M&T, 1992.

[24] P. Nunes, F. Marques, F. Pereira, and A. Gasull, "A contour-based approach to binary shape coding using a multiple grid chain code," *Signal Process.: Image Commun.*, vol. 15, no. 7–8, pp. 585–599, 2000.

[25] A. Rosenfeld and R. Klette, "Digital straightness," *Electron. Notes Theoret. Comput. Sci.*, vol. 46, pp. 1–32, 2001 [Online]. Available: http://www.elsevier.nl/locate/entcs/volume46.html

[26] P. Y. Simard, H. S. Malvar, J. Rinker, and E. Renshaw, "A foreground-background separation algorithm for image compression," in *Proc. Data Comp. Conf.*, Mar. 2004, pp. 498–507.

[27] H. Wang, G. M. Schuster, and A. K. Katsaggelos, "Object-based video compression scheme with optimal bit allocation among shape, motion and texture," in *Proc. Int. Conf. Image Processing*, Sep. 2003, pp. 785–788.

[28] H. Wang, G. M. Schuster, A. K. Katsaggelos, and T. N. Pappas, "An efficient rate-distortion optimal shape coding approach using a skeleton-based decomposition," *IEEE Trans. Image Process.*, vol. 12, no. 10, pp. 1181–1193, Oct. 2003.

[29] J. Ziv and A. Lempel, "Compression of individual sequences via variable-rate coding," *IEEE Trans. Inf. Theory*, vol. 24, no. 5, pp. 530–536, Sep. 1978.

Shankar Manuel Aghito (M'06) received the M.S. degree in telecommunications engineering from the University of Padova, Padova, Italy, in 2002, and the Ph.D. degree from the Technical University of Denmark, Lyngby, in 2006.

He currently holds a postdoctoral research position at the Research Center COM, Technical University of Denmark. His primary research interests include image and video compression.

Søren Forchhammer (M'03) received the M.S. degree in engineering and the Ph.D. degree from the Technical University of Denmark, Lyngby, in 1984 and 1988, respectively.

Currently, he is an Associate Professor with the Research Center COM, Technical University of Denmark, where is has been since 1988. His main interests include source coding, image and video compression, two-dimensional fields, and visual communications.